JSR 299: Contexts and Dependency Injection. Увод в JSR-303: Bean Validation.

Trayan Iliev

IPT – Intellectual Products & Technologies
e-mail: tiliev@iproduct.org
web: http://www.iproduct.org

JSR 299: Contexts and Dependency Injection (1)

- *Contexts and Dependency Injection for the Java™ EE platform* стартира през юни 2006 - JSR 299
- Финализирана е на 10 декември 2009
- Предоставя две фундаментални услуги необходими за реализация слабо-свързани компоненти с добре дефинран жизнен цикъл:
  - ***Lifecycle Contexts –*** осигуряват добре дефиниран жизнен цикъл за обекти запазващи вътрешно състояние, като освен стандартните е възможно дефиниране и на собствени контексти
  - ***Dependency Injection*** – възможността за ***typesafe*** "инжекция" на компоненти, декларативен избор на реализация по време на инсталиране (файл beans.xml)

JSR 299: Contexts and Dependency Injection (2)

- Поддръжка на модулност за Java EE компонентната архитектура, като взема в предвид зависимостите
- Интегрира се с ***Unified Expression Language (EL)*** за реферирane на обектите в даден контекст директно от ***Facelet*** или ***JSP*** страници
- Поддържа typesafe ***Interceptors*** към обектите и по този начин разделя ортогоналните функции и ги прилага към множество бийнове
- Чрез ***Decorators*** дава възможност за добавяме функционалност към вече съществуващи обекти
- Позволява прозрачна събитийно-ориентирана комуникация между отделните бийнове (***Events***)

JSR 299: Contexts and Dependency Injection (3)

- Добавя нов уеб ***Conversation*** контекст към стандартните (***Request***, ***Session*** и ***Application***), като позволява и добавяне на собствени контексти
- Осигурява пълен ***Service Provider Interface (SPI)***, който позволява на външни библиотеки лесно да се интегрират
- Елиминира нуждата от JNDI lookup базиран на символни имена и го заменя с ***typesafe инжекция*** с проверка на типовете по време на компилиране на програмата
- Силно редуцира нуждата от външни конфигурационни XML файлове, като ги заменя изцяло с ***анотации*** в кода, което позволява на инструменталните средства интроспкция по време на разработка

Кой обект е бийн според CDI?

- Според CDI бийн е контекстен обект, който дефинира (част от) състоянието и/или логиката на приложението
- Java EE клас е бийн, ако жизнения цикъл на неговите инстанции може да бъде управляван от контейнера съгласно контекстния модел на CDI
- Всеки бийн има:
  - непразно множество от типове
  - непразно множество от **квалификатори**
  - обхват (Request, Session, Application, Conversation, Dependent)
  - (незадължително) EL име
  - множество от интерцептори
  - реализация

Кои обекти са Managed Beans?

- Обекти дефинирани като Managed Beans от някоя Java EE спецификация (JSF, EJB)

  или

- изпълнява следните условия (без нужда от специална декларация):
  - не е нестатичен вътрешен клас
  - не е абстрактен или е анотиран като @Decorator
  - не е EJB
  - има конструктор без аргументи или такъв, чиито аргументи са анотирани с @Inject

Какви обекти можем да инжектираме с CDI?

- Инжектирането става с анотация ***@Inject*** приложена към полета, конструктори, методи или аргументи на методи
- Типове ресурси инжектируеми чрез CDI:
  - Почти всеки Java клас
  - Сесийни EJB
  - DataSources, JMS Topics, Queues Connection Factories
  - JPA Persistence Contexts (JPA EntityManager)
  - WebService референции
  - Референции към отдалечени интерфейси на EJB
- Можем да използваме **квалифициращи анотации** за да разграничим различните инстанции от един и същи тип в даден контекст – по подразбиране **@Default** квалификатор

Обхвати дефинирани от CDI

- **@RequestScoped** - конкретен HTTP request
- **@SessionScoped** - сесия на работа на потребителя с приложението
- **@ApplicationScoped** - споделен обхват от всички потребители на конкретното приложение за всички заявки
- **@ConversationScoped** – дефиниран експлицитно от разработчика по програмен път – може да обхваща една или повече заявки с общо **conversation Id (cid)** (long runnig conversations) не може да излиза извън рамките на сесията
- **@Dependent** - обхват по подразбиране, при който се създава по една уникална инстанция на инжектираните бийнове за всеки обект, който ги инжектира, в неговия обхват

Ключови анотации в CDI (javax.inject)

- @Inject – Обозначава инжектируеми конструктори, методи и полета
- @Named – Позволява да дефинираме EL имена на Managed Beans за JSF
- @Qualifier – Обозначава квалифицираща анотация
- @Singleton – Обозначава тип, от който инжектора трябва да създаде само една инстанция

Ключови анотации в CDI (javax.enterprise.inject)

- @Alternative – Специфицира че бийнът е алтернатива
- @Any – Всеки бийн има квалификатор @Any, освен специалните бийнове с квалификатор @New
- @Default – Тип квалификатор по подразбиране
- @Disposes – Обозначава инжектируемия параметър, съдържащ обекта, от който ще се освобождаваме в **disposer method**
- @Model = @Named + @RequestScoped (JSF)
- @New – Позволява да получим нова инстанция на бийна

Ключови анотации в CDI (javax.enterprise.inject)

- @Produces – Обозначава метод или поле **Producer**
- @Specializes – Индикира, че един бийн директно **специализира** друг бийн
- @Stereotype – Специфицира, че съответния тип анотация е **стереотип**
- @Typed – Ограничава типвете на бийна, от които той може да се инжектира

Trayan Iliev
IPT – Intellectual Products & Technologies Ltd.

Използване на CDI в Java™ EE 6

- Примери ...

Избор на алтернативни реализации при CDI

- **@Alternative** – дава възможност за избор на подходяща алтернатива, която реализира същия интерфейс (например за целите на тестването) декларативно – във файла **beans.xml** :

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">
    <alternatives>
        <class>mybeans.FormatterTestImpl</class>
    </alternatives>
</beans>
```

Специализация при CDI

- **@Specializes –** позволява един по-специализиран бийн изцяло да замести друг по време на изпълнение (специализирания бийн е наследник на базовия) :

```
@Specializes public class BetterTaskImpl extends TaskImpl
{
    .....
}
```

Използване на продуциращи полета, методи и конструктори при CDI

- **@Produces –** позволява да инжектираме обекти, които не са бийнове или чиято инициализация и/или конкретен тип могат да варират - например :

```
@Produces
@ChosenFormatter
@RequestScoped
public Formatter getFormatter(@New FormatterTestImpl fti,
        @New FormatterImpl fi) {
    switch (chooseFormatter) {
        case 1:   return fti;
        case 2:   return fi;
    }
}
```

По-сложни елементи: Events, Interceptors, Decorators

- **Events:** 1) дефинира се клас за събитието с един или повече квалификатори; 2) анотация **@Observes** + квалификатори ни позволява да регистрираме слушатели на събития; 3) чрез инжектиране на интерфейса **Event<T>** и извикване на метода му **fire(eventPayload)** създаваме събитие.
- **Interceptors**: анотации **@InterceptorBinding** и **@Interceptor** + включване на интерцептора в елемента **<interceptors>** на **beans.xml**, за многократно използване
- **Decorators**: **анотации @Decorator** и **@Delegate**, за разширяване на функционалността на конкретен клас

Weld Servlet Injection - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/ipt-servlet-event/

Google

Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Ser...

Add a new product and price:

Product Name:	Thinking In Java	Required
Product Price:	25.00	Required

Submit Reset

CDI_IPT_Iliev.od... ipt-servlet-event ... Inbox - tiliev@ipr... Weld Servlet Inje... Weld Servlet Inje... C:\Business\IPT_... EN 12:47 PM

Weld Servlet Injection - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/ipt-servlet-event/

Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Servlet... Weld Ser...

Add Product Servlet

Demonstrates Use of CDI

Add a new product and price:

Product Name:	Thinking In Java	Required
Product Price:	25.00	Required

Submit Reset

Product Successfully Added: Thinking In Java

CDI_IPT_Iliev.od... ipt-servlet-event ... Inbox - tiliev@ipr... Weld Servlet Inje... C:\Business\IPT_... IPT-Events.png-1... EN 12:52 PM

CDI Events – дефиниране на Event клас

```
package org.iproduct.eshop;

public class NewProductAdded {
    private String productName;

    public NewProductAdded(String productName) {
        this.productName = productName;
    }
}
```

CDI Events – @Inject Event<T>, fire(eventData)

```
@Inject @Any
 Event<NewProductAdded> newProductEvent;
 private boolean empty = true;
 public void addProduct() {
     if ((pData.getProductName() != null
             && pData.getProductName().length() > 0)
             && (pData.getProductPrice() > 0)) {
         empty = false;
         // Fire an event
         newProductEvent.fire(
             new NewProductAdded(pData.getProductName()));
     }
 }
```

CDI Events – Event Listeners: @Observes

```
public void procuctAdded(
    @Observes NewProductAdded event) {
    System.out.println("----> ProductAddedEvent catched: " +
          event);
}
```

Trayan Iliev
IPT – Intellectual Products & Technologies Ltd.

Interceptors – @InterceptorBinding анотация

```
@Inherited
@InterceptorBinding
@Retention(RUNTIME)
@Target({METHOD, TYPE})
public @interface Logged {
}
```

Interceptors – @Interceptor

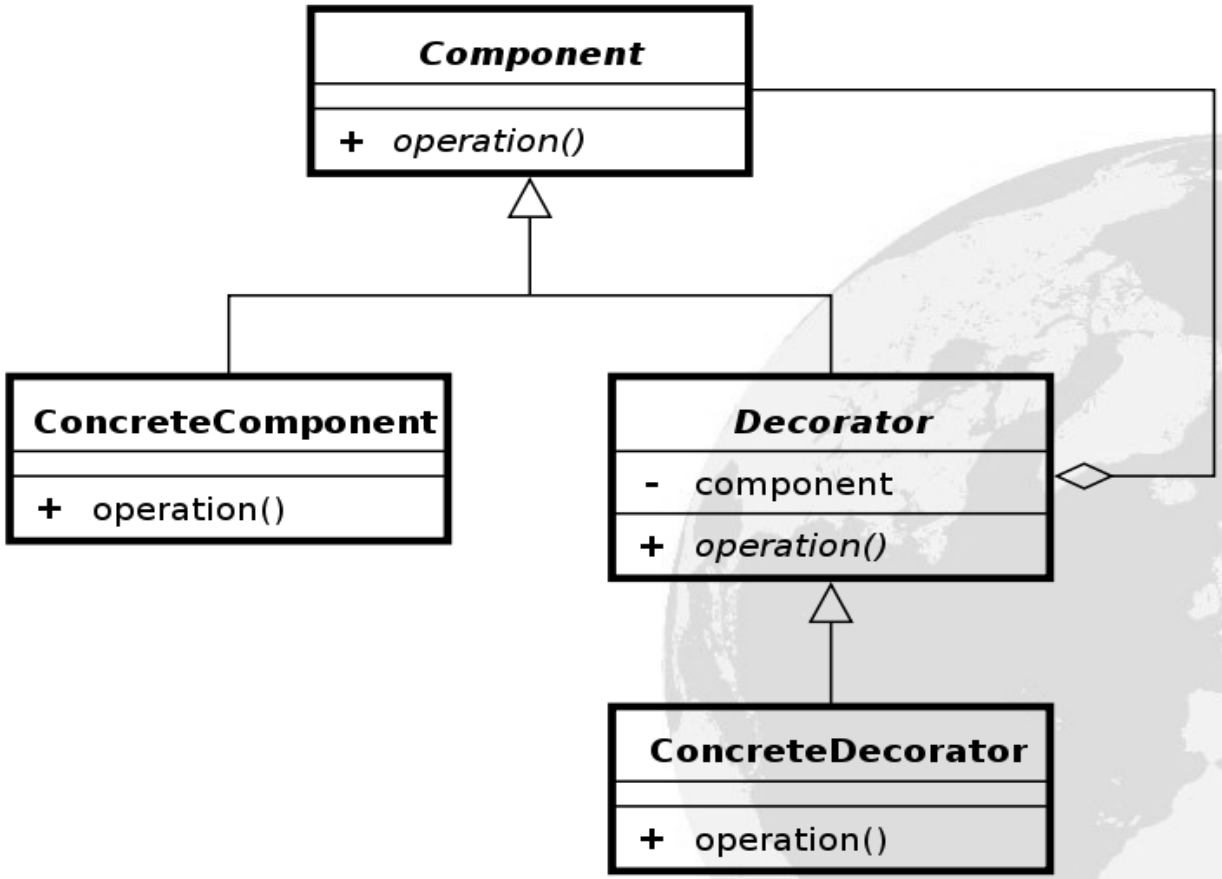
```
@Logged @Interceptor
public class LoggedInterceptor implements Serializable {
  private Logger logger =
      Logger.getLogger("org.iproduct.eshop.interceptor");
  @AroundInvoke
  public Object logMethodEntry(InvocationContext invContext)
          throws Exception {
      logger.info("Entering method: "
 + invocationContext.getMethod().getName() + " in class "
 + invocationContext.getMethod().getDeclaringClass().getName());
      return invocationContext.proceed();
  }
}
```

Interceptors – <interceptors> в beans.xml

```
<beans xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">
    <interceptors>
       <class>billpayment.interceptor.LoggedInterceptor</class>
    </interceptors>
</beans>
```

```
@Logged
@Named @ApplicationScoped @Default
public class ProductController implements Serializable {
   ...
```

Reusable Design Pattern: Decorator



Source: http://en.wikipedia.org/wiki/File:Decorator_UML_class_diagram.svg
License: public domain

Decorators – @Decorator и @Delagate

```
@Decorator
public abstract class FormatterDecorator implements Formatter {
  @Inject
  @Delegate
  @Any
  Formatter delegate;
  public static final Logger logger =
        Logger.getLogger("org.iproduct.eshop.decorators");
  @Override
  public String formatString(String s) {
     logger.log(Level.INFO, "Entering decorator method.");
     return delegate.formatString(s).toUpperCase();
  }
}
```

Decorators – <decorators> в beans.xml

```
<beans xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">
      <decorators>
            <class>decorators.CoderDecorator</class>
      </decorators>
</beans>
```

JSR-303: Bean Validation (1)

- *Bean Validation* стартира през юли 2006 - JSR 303
- Финализирана е на 16 ноември 2009
- Валидацията е обща задача, която се осъществява през всички слоеве на приложението – от презентационния до персистирането на данните
- Често една и съща логика за валидация се реализира многократно във всеки слой, което води до чести грешки е несъответствия, както и до дублиране на усилия
- За да се справят с проблема, често разработчиците кодират валидационната логика директно в домейн модела, което води до смесване на бизнес логика и метаданни за валидиране на отделните свойства

JSR-303: Bean Validation (2)

- *JSR 303: Bean Validation* предлага набор от стандартни **ограничения (constraints)** относно данните, под формата на **анотации**, които обозначават **полета, методи или класове** на JavaBean компоненти, като например JPA Entities или JSF Managed Beans
- Има множество **предварително дефинирани** анотации, както и **възможност за създаване на собствени** такива и свързването им с клас, който да реализира валидационната логика
- Вградените анотации са дефинирани в пакет **javax.validation.constraints**

Bean Validation – вградени анотации (1):

- **@AssertFalse –** елемент от булев тип трябва да е лъжа
- **@AssertTrue –** елемент от булев тип трябва да е истина
- **@Min, @DecimalMin –** минимална стойност на елемент от числов тип
- **@Max, @DecimalMax –** максимална стойност на елемент от числов тип
- **@Digits** – атрибути fraction и integer за дробната и цялата част на елемент от числов тип
- **@Future** – валидиране на бъдеща дата (Date и Calendar)
- **@Past** – валидиране на минала дата (Date и Calendar)
- **@Size** – min и max размер на String, Collection, Map или Array

Bean Validation – вградени анотации (2):

- **@NotNull** – елементът трябва да е различен от **null**
- **@Null** – елементът трябва е **null**
- **@Pattern** – елементът трябва да съответствува на посочения в атрибута **regexp** регулярен израз
- **@Valid** – анотация в пакета **javax.validation**, която указва, че трябва да се извърши рекурсивна валидация на всички обекти свързани с посочения обект
- Възможно е създаване на нови собствени анотации и композитни анотации с използване на **@Constraint, @GroupSequence, @ReportAsSingleViolation, @OverridesAttribute**

Пример за Bean Validation анотации:

```
public class Email {
  @NotEmpty @Pattern(".+@.+\\.[a-z]+")
  private String from;
  @NotEmpty @Pattern(".+@.+\\.[a-z]+")
  private String to;
  @NotEmpty
  private String subject;
  @Min(1) @Max(10)
  private Integer priority;
  @NotEmpty
  private String body;
      }
```

Източник: Javadoc Documentation /api/javax/persistence/CollectionTable.html

Bean Validation – собствена анотация:

```
@Size(min=4, max=4)
@ConstraintValidator(validatedBy = PostCodeValidator.class)
@Documented
@Target({ANNOTATION_TYPE, METHOD, FIELD})
@Retention(RUNTIME)
public @interface PostCode {
  public abstract String message() default
                        "{package.name.PostCode.message}";
  public abstract Class<?>[] groups() default {};
  public abstract Class<? extends ConstraintPayload>[]
                        payload() default {};
```

Източник: Javadoc Documentation /api/javax/persistence/CollectionTable.html

Bean Validation – клас PostCodeValidator

```
public class PostCodeValidator implements
                                ConstraintValidator<PostCode, String> {
    private final static Pattern POSTCODE_PATTERN =
                                Pattern.compile("\\d{4}");
    public void initialize(PostCode constraintAnnotation) { }
    public boolean isValid(String value,
                                ConstraintValidatorContext context) {
        return POSTCODE_PATTERN.matcher(value).matches();
    }
}
```

Източник: Javadoc Documentation /api/javax/persistence/CollectionTable.html

Bean Validation – композитна анотация:

```
@ConstraintValidator(validatedBy = {})   @Documented
@Target({ANNOTATION_TYPE, METHOD, FIELD})
@Retention(RUNTIME)
@Pattern(regexp = "\\d{4}")
@ReportAsSingleViolation
public @interface PostCode {
  public abstract String message() default
                          "{package.name.PostCode.message}";
  public abstract Class<?>[] groups() default {};
  public abstract Class<? extends ConstraintPayload>[]
                          payload() default {};
```

Източник: Javadoc Documentation /api/javax/persistence/CollectionTable.html

References

- JSR 299: Contexts and Dependency Injection for the JavaTM EE platform – http://jcp.org/en/jsr/detail?id=299
- JSR 330: Dependency Injection for Java – http://jcp.org/en/jsr/detail?id=330
- JSR 303: Bean Validation –

  http://jcp.org/en/jsr/detail?id=303
- JSR 299: Contexts and Dependency Injection for the JavaTM EE platform – http://jcp.org/en/jsr/detail?id=299
- Java EE 6 Tutorial – http://java.sun.com/javaee/6/docs/tutorial/doc/
- Sang Shin's Java Passion educational web site – http://www.javapassion.com/

Благодаря за вниманието!

Въпроси?